VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2018 Branislav Bellovič

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Branislav Bellovič**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: SDE - Software Solutions s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **RNDr. Eliška Ochodková, Ph.D.**

Konzultant bakalářské práce: RNDr. Jaroslav Žáček, Ph.D.

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24.dubna 2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB – TU Ostrava.

V Ostravě 20.dubna 2018

SDE Software Solutions s.r.o.
Dornych 90, 617 00 Brno
IČ: 253 09 978
tel.: 545 211 280
(4)

.....

Na tomto mieste by som rád poďakoval RNDr. Eliške Ochodkovej, Ph.D za vedenie tejto práce a pomoc pri jej zhotovení. Ďalej by som sa chcel poďakovať RNDr. Jaroslavovi Žáčkovi Ph.D a Ondřejovi Patočkovi za konzultácie a pomoc zo strany firmy SDE Software Solutions. Moje poďakovanie patrí tiež kolektívu firmy SDE Software Solutions, ktorý tvoril príjemné prostredie pre zhotovenie tejto práce.

Abstrakt

Tento dokument popisuje moju bakalársku prax vykonanú vo firme SDE Software Solutions, ktorá patrí k outsourcingovým firmám. V rámci firmy som bol zaradený do tímu pracujúceho pre firmu Velleros sídliacu v Severnej Karolíne, USA. Podieľal som sa na tvorbe softwarového riešenia zabezpečujúceho včasne varovanie pred nebezpečenstvami ohrozujúcimi život. Toto riešenie zahŕňa meranie a monitorovanie pokrytia signálu v USA.

Kľúčové slová: prax, systém včasného varovania, monitorovanie pokrytia signálu

Abstract

This document describes my intership at SDE Software Solutions, which belongs to outsourcing companies. I was part of the team working for Velleros situated in North Carolina, USA. I was involved in creating a software solution that provides early warning of life-threatening dangers. This solution includes measurement and monitoring of signal coverage in the USA.

Key Words: intership, early warning system, network signal monitoring

Obsah

Seznam použitých zkratek a symbolů

LTE	–	Long Term Evolution
CDMA	–	Code-division multiple access
FEMA	–	Federal Emergency Management Agency
BTS	–	Base transceiver station
ECI	–	E-UTRAN Cell Identifier
eNodeB	–	Evolved Node B
VGO	–	Velleros geographical object
UX	–	User experience
HTML	–	Hypertext Markup Language
CPU	–	Central processing unit
CMAS	–	Commercial Mobile Alert System
OMC	–	Operation and Maintenance Center

Zoznam obrázkov

Seznam výpisů zdrojového kódu

1 Úvod

Tento dokument vznikol v priebehu odbornej praxe v rámci výuky na na Technickej Univerzite Ostrava a popisuje jej priebeh.

Prvá časť dokumentu v skratke popisuje firmu SDE Software Solutions,v ktorej som odbornú prax vykonal. V širšom rozsahu je opísaná firma Velleros, s ktorou firma SDE Software Solutions spolupracuje a projekt, na ktorý som bol zaradený v rámci spolupráce firiem. Hlavná časť dokumentu je zameraná na úlohy, ktoré som vypracoval počas odbornej praxe. Túto časť som rozdelil na dva celky.

Prvý celok popisuje úlohy, ktoré boli zamerané predovšetkým na zlepšenie skriptu pre nahrávanie zákazníckych dát do databázy a jeho prispôsobenie pre podporu LTE sieti. Úlohy prepojené v tomto celkou boli z veľkej miery písane v jazyku PHP, doplnené jazykmi HTML a JavaScript.

Druhá časť vypracovaných úloh bola zameraná na vizualizáciu zákazníckych dát a taktiež na rozšírene o podporu LTE sieti. Cieľom týchto úloh bolo predovšetkým zlepšenie celkového užívateľského dojmu pri používaní mapy, ktorá na vizualizáciu dát slúžila. Úlohy boli vypracované v JavaScripte so značným využitím JavaScriptového rozšírenia OpenLayers.

V závere dokumentu sa nachádzajú informácie o praktických, ale ja teoretických vedomostiach nadobudnutých počas štúdia a využitých pri vykonávaní bakalárskej praxe. Úplný záver popisuje chýbajúce vedomosti, ktoré som sa musel naučiť pre úspešné vykonanie praxe a celkové zhrnutie priebehu praxe.

2 O Firme

SDE Software Solutions je česká firma založená v roku 1995 v Brne. Firma má v súčastnosti okolo 90 zamestnancov pracujúcich v pobočkách v Brne a Ostrave.

Ostravská pobočka sa skladá zhruba z 25 zamestnancov. Pobočka je situovaná v podnikateľskom inkubátore, ktorý je pod správou Vysoké školy báňské – Technické univerzity Ostrava s ktorou firma spolupracuje.

Obr. 1: SDE Software Solutions Logo

2.1 Štruktúra firmy

Firma patrí k outsourcingovým[1] firmám a je zameraná na americký trh.

Tímy sa zvyčajne skladajú z 2-6 programátorov a pracujú na projektoch z rôznych oblasti. Spektrum programátorských jazykov vo firme je rovnako rôznorodé ako projekty. Najčastejšie používane sú PHP, C-Sharp a Ruby On Rails na strane serveru a JavaScriptove frameworky ako React a Angular na strane klienta.

2.2 Velleros

V rámci firmy som bol zaradený do 2 členného tímu pracujúceho pre firmu Velleros. Velleros je firma situovaná v Severnej Karoline. Jej softwarové produkty sú určene predovšetkým pre mobilných operátorov a ich zákazníkov.

Softwarové produkty tejto firmy sa zameriavajú predovšekym na distribúciu upozornení pred nebezpečenstvami ohrozujúcimi život. Sprostredkúvajú napríklad upozornenia vydane úradom FEMA [9], ale aj civilné ohrozenia zo strany zločincov.

2.2.1 Priebeh práce vo Vellerose

Tak ako väčšina firiem má aj firma Velleros niekoľko zaužívaných procesov uľahčujúcich vývoj, rozdelenie práce a integráciu zmien. Aby som sa mohol plne zapojiť do práce niektoré z týchto procesov som si musel naštudovať, iné som si osvojil počas vývoja.

[1] Outsourcing znamená, že firma vyčlení rôzne podporné a vednejšie činnosti a zverí ich zmluvne inej spoločnosti čiže subkontraktovi, špecializovanému na príslušnú činnosť.

2.2.2 Agilný prístup

Pre riadenie vývoja používa firma agilnú metodiku z názvom SCRUM [4]. Vzhľadom k veľkosti tímu, ktorý sa skladal z 5 členov nebolo možné striktne dodržiavať všetky aspekty tohto prístupu k vývoju. Tím sa skladá z product ownera, scrum mastera, a troch programátorov. Denne mítingy prebiehali pravidelne s využitím platformy Google Hangouts [5]. Keďže oproti USA sme v 6 hodinovom časovom posune, ostravský tím mítingom končil deň, zatiaľ čo americký tím ním deň začínal. V rámci mítingu sme obvykle hlásili čo sme robili za posledný deň, poprípade sme diskutovali problémy spojené s projektom. Americký tím nám tiež hlásil ich pokrok za predchádzajúci deň. Mojim veľkým prekvapením bola ústretovosť a ochota pomôcť aj nad rámec pracovných povinnosti zo strany amerického, ale aj českého tímu.

Veľkou prekážkou však bola pre mňa takzvaná americká angličtina, ktorá mi komplikovala komunikáciu počas celého priebehu praxe.

2.2.3 Správa a plánovanie projektu

Pre plánovanie sprintu sme používali platformu Pivotal Tracker [6], ktorá mala zabudovanú integráciu s GitHubom [7], čo bolo naše primárne úložisko všetkých projektov. So samotným Pivotal Trackerom som sa zoznámil veľmi rýchlo. Moju aktivitu tu predstavovalo vyberanie si úloh z tzv. backlogu a nastavenie časového odhadu nutného pre splnenie danej úlohy. S odhadom mi zo začiatku pomáhal kolega z ostravského tímu, ktorý mal väčšie skúsenosti s projektom. Súčasť scrumu, ktorá sa dodržiavala v menšej miere boli demá, ktoré sa uskutočňovali niekedy aj v 1 a pol mesačných odstupoch, čiže som sa veľmi dem nezúčastňoval. Pivotal tracker sme tiež využívali pre technickú komunikáciu. Každá priradená úloha mala možnosť vlastnej diskusie, kde sme obvykle písali poznámky, respektíve otázky, na ktoré odpovedali členovia amerického tímu.

2.2.4 Správa verzii kódu

Ako som už spomínal na správu zdrojobvého kódu sme používali git [8] spolu so známym nástrojom GitHubom. S gitom ani Githubom som pred praxou nepracoval a preto som si musel naštudovať ako fungujú a ako spolupracujú. Git som zvyčajne nepoužíval z príkazoveho riadku, ale po odporúčaní ostrasvkého kolegu som si stiahol voľne dostupný nástroj SourceTree [10], ktorý som neskôr začal používať aj na vlastných, predovšetkým školských projektoch.

Pre každú pridelenú úlohu som si väčšinou pomocou SourceTree vytvoril vlastnú *branch*, ktorej názov vystihoval danú úlohu. Keďže Pivota tracker podporoval integráciu s GitHubom ku každému *commitu* som pridával identifikátor danej úlohy. Odkaz na tieto *commity* bol potom pridaný do komentárov k danej úlohe v Pivotal Trackeri, a tým uľahčoval orientáciu. V prípade, že *commit* ukončoval úlohu pridal som značku [Finishes hash-id] a úloha sa automaticky označila ako dokončená a niekto z amerického tímu ju mohol otestovať a pridať do nasledujúcej verzie programu.

Pred finálnym *commitom* som ešte väčšinou pomocou GitHubu požiadal o tzv. *„pull request“* ostravského kolegu, ktorý kód prekontroloval a v prípade nutnosti okomentoval. Následne sme väčšinou v kancelárii prediskutovali možne riešenie, respektíve prečo som zvolil daný postup. Kontrola kódu fungovala v oboch smeroch, čiže ja som schvaľoval *„pull requesty“* ostravskému kolegovi. Po zaradení mojich zmien do vetvy s názvom develop som mohol svoju vetvu vymazať a začať pracovať na novej úlohe.

2.3 OMC

Webová aplikácia **O**peration and **M**aintenance **C**enter je jedným z produktov firmy Velleros. Keďže som na tomto systéme vykonal niekoľko úloh, rad by som ho priblížil. Systém je určený telekomunikačným spoločnostiam, ktoré si zakúpia jednu zo služieb Vellerosu. Výsledný poskytovaný obsah záleží od zakúpených služieb. V rámci väčšiny služieb existuje niekoľko užívateľských rolí.

Jedna z najväčších časti OMC slúži na nahrávanie dát s informáciami o anténach pre rôzne telekomunikačné technológie. Vďaka informáciám z týchto dát ďalšie služby dokážu odosielať dôležité informácie na mobilné telefóny zaregistrované k danej anténe. K týmto informáciám patri napríklad upozornenia pred katastrofou alebo zločincom. Systém OMC ďalej umožňuje vizualizáciu aktuálnej výstrahy a umiestnenie antén na mape. Ďalšou funkciou tejto služby je história upozornení.

Druha veľká časť je prepojená s mobilnou aplikáciou vyvíjanou Vellerosom. Táto služba umožňuje bohato konfigurovateľnú vizualizáciou dát zozbieraných z mobilnej aplikácie, pomocou mapy alebo rôznych grafov. Tieto dáta slúžia pre monitorovanie dostupnosti signálu a analýzu dát za účelom lepšieho pokrytia slabo pokrytých oblasti.

2.3.1 Technológie použite v OMC

Tento systém je vyvíjaný už viac ako 7 rokov. Pre realizáciu webovej aplikácie bol zvolený jazyk PHP verzie 5.3.3 s nadstavbou MVC frameworku s nazvom Zend Framework [11] . Používaná verzia 1.11 už nie je niekoľko rokov podporovaná a potenciál tohto frameworku je v projekte využívaný v malej miere a s rozporom voči všeobecnej myšlienke MVC patternu. Tento stav ma často obmedzoval pri rozhodnutiach týkajúcich sa návrhu a implementácie. Návrh riešenia úloh, ktoré mi boli zadané by vyžadovali veľkú mieru zásahu do už existujúceho kódu . Tento čas by však pravdepodobne presiahol pridelený čas a preto som sa musel uspokojiť s kompromisom.

3 Vypracované úlohy spojené s upload scriptom

3.1 Validácia hlavičky importovaného CSV súboru

Moja prvá úloha bola spojená s kontrolou hlavičky súboru. Hlavička súboru je obvyklé prvý riadok obsahujúci stĺpce s názvami jednotlivých parametrov antény. Mojou úlohou bol zabezpečiť aby sa skontrolovala prítomnosť týchto stĺpcov, prípadne ich správne umiestnenie. Algoritmus mal pre-definovane pozície týchto stĺpcov, respektíve názvov týchto atribútov a rátal s tým, že dáta budú vždy v správnom stĺpci. Táto implementácia nerátala s ľudskou chybou, kvôli ktorej import dát nečakane zlyhával.

3.1.1 Návrh riešenia problému

Po analýze vstupných dát som sa rozhodol rozdeliť hlavičku na dve časti:

1. Všeobecná časť, ktorú obsahoval každý záznam.
2. Časť, ktorá bola špecifická pre danú technológiu.

Algoritmus pre nahrávanie dát podporoval 2 formáty, ktoré rozlišoval podľa prvého atribútu hlavičky. Pre formát e911 bola podporovaná iba technológia CDMA, preto toto delenie nebolo potrebné a validácia mohla zakaždým prebehnúť pre všetke atribúty.

Pre formát velleros som sa najskôr rozhodol kontrolovať len všeobecnú časť hlavičky a pri narazení na prvý záznam s špecifickou technológiou kontrolovať časť špecifickú pre túto technológiu. Vďaka tomuto rozdeleniu nemôže dôjsť k zlyhaniu importu, ak chýba atribút technológie, ktorá sa nenachádza v danom súbore. Taktiež predpokladám zníženie trvania importu, ktorého trvanie som však neoveroval, pretože pri menšom počet opakovaní rovnakej operácie je menšia časová náročnosť evidentná.

3.1.2 Implementácia

Realizáciu som sa rozhodol začať formátom Velleros. Vytvoril som si preto tri funkcie:

1. validateCommonHeaders
2. validateCdmaHeaders
3. validateLteHeaders

Každá funkcia obsahovala 2 vstupné parametre:

1. key-value pole A obsahujúce hlavičkový súbor
2. pole B obsahujúce preddefinované názvy a pozície atribútov

Algoritmus postupne zisťuje a zapisuje si či sú prítomné všetky atribúty a či ich pozícia je správna. Ak je pozícia nesprávna, tak sa aktualizuje v poly B. Následne sa skontroluje prítomnosť všetkých atribútov definovaných v poly B. Ak nejaké atribúty chýbajú, import dát sa preruší s chybovou hláskou obsahujúcou názvy chýbajúcich atribútov. Inak funkcia vráti aktualizované pole s pozíciami atribútov.

Pri následnej implementácii funkcionality pre format e911 som sa odklonil od návrhu a namiesto vytvorenia jednej funkcie pre všetky atribúty som rozšíril vytvorené funkcie pre format velleros o parameter format, na základe ktorého sa skontrolovali správne atribúty. Funkčnosť algoritmu som otestoval ručne pomocou ukážkových súborov.

3.1.3 Čas vypracovania

Zoznámenie sa s firmou a pracovnými postupmi mi trvalo zhruba prvé 4 dni. Ďalej som sa musel zoznámiť so štruktúrou projektu a vypracovanie prvej úlohy mi zabralo 5 dni.

3.2 Validácia LTE záznamov

Skript pre nahrávanie dát zákazníkov obsahoval kontrolu prítomnosti niektorých atribútov pre CDMA záznamy. Mojou úlohou bolo implementovať rovnakú funkcionalitu pre LTE záznamy. Návrh riešenia preto nebol potrebný.

Atribúty sa rozdeľujú na povinné a nepovinné. Pre kontrolu prítomnosti hodnoty pre daný atribút som použil rovnakú funkciu aká bola použitá pre kontrolu CDMA záznamov. Táto funkcia kontrolovala či je vstupný parameter prítomný alebo neobsahuje len neviditeľné znaky. Ak sa našiel nejaký prázdny atribút import skončil s chybovou hláskou.

3.2.1 Implementácia

Pre samotnú kontrolu som implementoval funkciu, ktorá postupne kontrolovalá povinné aj nepovinné atribúty pomocou regulárnych výrazov, alebo pre jednoduchšie overenia zabudované funkcie jazyka PHP jazyka [12]. Pre tvorbu jednotlivých výrazov som využil akademické vlastnosti a officialnu dokumentáciu . Pre urýchlenie a uľahčenie práce som použil online nástroj pre tvorbu regulárnych výrazov [13].

```
if (!preg_match("/^0[xX][0-9A-Fa-f]{1,7}$|^[0-9]+$/", $info["eci"])) {
    throw new Error("Invalid ECI (eNodeB+Sector) format on line $line:
      {$info["eci"]}. ECI must decimal or hexadecimal number");
}
```

Výpis 1: V ukážke zdrojového kódu môžme vidieť kontrolu atribútu *eci* pomocou regulárneho výrazu. Ak hodnota *eci* nie je decimálne alebo hexadecimálne riadok sa vyhodnotí ako neplatný.

3.2.2 Čas vypracovania

V tejto úlohe som prvé 2 dni strávil úpravou kódu a rozdelením do menších časti. Následujúce 2 dni som strávil samotnou implementáciou.

3.3 Rozšírená kontrola ECI identifikátora pre LTE záznamy

Pôvodný predpoklad, že hodnota atribútu ECI sa vždy bude rovnať hodnote atribútu eNodeB sa po diskusii so zákazníkom vyvrátil. Z diskusie so zákazníkom taktiež vyplynulo že hodnota atribútu ECI sa dá vypočítať a preto nemusí byť vždy prítomná v súbore s dátami určenými pre import. Úloha obsahovala niekoľko bodov:

1. hodnota atribútu ECI nie je ďalej povinná
2. unikátne meno pre databázový objekt bude obsahovať časť meno + „ ECI “ + hodnota ECI namiesto meno + „ CellID “ + hodnota cellid/eNodeB. Hodnota ECI bude vypočítaná ako eNodeB * 256 + číslo sektora antény.
3. Hodnota ECI v sekcii <data> bude vypočítaná rovnakým vzorcom v prípade, že zadaná hodnota pre atribút ECI chýba.
   (a) Ak atribút ECI hodnotu obsahuje, použitá bude zadná hodnota. Ak sa zadná hodnota líši od vypočítanej hodnoty, informácia o rozdielnych hodnotách bude zobrazená ako upozornenie po prevedení skriptu. Výsledný text by mál vyzerať napríklad nasledovne "10 vypočítaných hodnôt zo 100 sa nezhoduje so zadanými hodnotami"

3.3.1 Analýza a návrh

Prvá časť úlohy mi nezabrala veľa času. Keďže atribút ECI nebol ďalej povinný tak som odstránil striktnú kontrolu prítomnosti pomocou vyššie zmienenej funkcie za jednoduchú podmienku ktorá vykonala kontrolu pomocou regulárneho výrazu len v prípade že tento atribút bol zadaný.

Bod číslo 2 a 3 som musel diskutovať s vedúcim programátorom z USA. Mojou úlohou dodržiavať firmou stanovený formát **VGO**, ktorý sa používa na výmenu údajov medzi službami v rámci firmy ako aj na komunikáciu so spolupracujúcimi službami tretích strán. Do tohto formátu sa po kontrole a zoskupení prevádzali aj dáta zo skriptu pre nahrávanie zákazníckych dát pred následným odoslaním na ďalší server, ktorý sa staral o samotné uloženie do MySQL databáze a ďalšími úlohami spojenými s dátami.

```
<?xml version='1.0'?>
<vgol xmlns='vgol:1.0'>
  <vgo>
    <layer>Organization 1</layer>
    <name>Ostrava2: ECI 6648834 on 192.168.1.54</name>
```

```
      <type>LTE</type>
      <data>
          <mme_ip_address>192.168.1.54</mme_ip_address>
          <mme_port>34452</mme_port>
          <eci>6648834</eci>
          <eNodeB>25972</eNodeB>
          <sector_id>2</sector_id>
          <operator_mccmnc>3110444</operator_mccmnc>
          <mme_vendor>Huawei</mme_vendor>
          <tac_id>20</tac_id>
      </data>
      <area>
         <sector>18.342219,49.435030 90 60 29</sector>
      </area>
      <sectorinfo>
      <sector>
         <id>2</id>
         <lon>18.342219</lon>
         <lat>49.435030</lat>
         <azimuth>90</azimuth>
         <width>60</width>
         <radius>29000</radius>
         <units>Meters</units>
      </sector>
      </sectorinfo>
   </vgo>
</vgol>
```

Výpis 2: Ukážka zobrazuje anténu serializovanú do XML formátu VGO

Z ukážky zdrojového kódu 2 môžeme pozorovať, že údaje pre CDMA sieť obsahujú všetky sektory v jednom zázname typu VGO a že v časti <data> sa nachádza jeden záznam s názvom *cellid*. Zo zadania úlohy vyplýva, že ak bude mať daná anténa viac ako jeden sektor, vznikne viac potenciálnych hodnôt pre meno databázového objektu a pre položku eNodeB v časti dáta. Po diskusii a následnej analýze dopadov na systém som sa rozhodol pre návrh, v ktorom bude každý sektor LTE antény uložený ako separátny záznam v databáze. Aby som dosiahol tento efekt zmenil som respektíve som zjednodušil finálne spracovanie dát do formátu VGO, ktoré pôvodne pracovalo so štruktúrou viacerých sektorov v jednom VGO zázname.

Pri analýze zadania som došiel k názoru že výpis počtu nesprávne zadaných hodnôt ECI nemá dostatočný prínos a že by bolo vhodnejšie poskytnúť údaj o riadku v csv súbore, kde sa

nesprávna hodnota vyskytla. Tento návrh bol schválený.

3.3.2 Implementácia

Samotnú kontrolu som umiestnil do časti algoritmu ktorá prevádzala csv riadok do php objektu z dôvodu prítomnosti aktuálne spracovávaného riadka. Kontrola spočívala v jednoduchom porovnaní vypočítanej a zadanej hodnoty. Ak sa dané hodnoty líšia, číslo riadka sa uloží do poľa určeného pre nesprávne hodnoty ECI. Ak toto pole nie je prázdne, tak sa po úspešnom dokončení nahrávania dát vypíše hláška s riadkami obsahujúcimi nesprávne vypočítané hodnoty ECI. Keďže súbor môže obsahovať veľké množstvo nesprávnych hodnôt použil som HTML element <summary>, vďaka ktorému je užívateľ informovaný o výskyte nesprávnych hodnôt a až po kliknutí na tento element vidí riadky s konkrétnymi hodnotami.

3.3.3 Čas vypracovania

Vypracovanie tejto úlohy mi trvalo okolo 4 dní.

3.4 Duplikáte záznamy v CSV súbore

Na úvod by som rád poznamenal, že ďalej opisované úlohy mi boli pridelené postupne a často vyplývali z požiadaviek plynúcich z používania softvéru zákazníkmi firmy Velleros. Preto problém, návrh a riešenie bude uvádzať ako výsledný generalizovaný stav po dokončení poslednej úlohy spojenej s touto problematikou.

3.4.1 Motivácia

Tento problém vyplýval z predpokladu že každá anténa ma unikátny identifikátor, ktorý je v prípade LTE antén ECI a v prípade CDMA antén cellid. Toto spôsobovalo že skript pre nahrávanie dát prepísal niektoré hodnoty. Čiže ak sa v jednom súbore nachádzali 2 antény s rovnakým ECI, respektíve cellid do databázy bola uložená len hodnota, ktorá sa nachádzala na neskoršej pozícii.

Mojou úlohou bolo zistiť čo spôsobuje toto správanie, následne zaistiť uloženie všetkých záznamov z CSV súboru a v prípade duplicitných záznamov poskytnúť chybovú správu s riadkom, kde sa nachádzal nejaký duplicitný riadok, podobne ako pre nesprávne hodnoty ECI.

3.4.2 Analýza a Návrh

V prvom kroku som sa zameral na lokalizáciu miesta kde sa duplicitné záznamy prepisujú. Napriek faktu, že postup algoritmu pravdepodobne vyplýva z predchádzajúcich úloh pre lepšiu orientáciu ho načrtnem ešte raz.

1. Otvorí sa súbor a načíta sa do pamäte.

2. Postupne sa načítavajú riadky až kým algoritmus nenarazí na hlavičkový riadok s názvami stĺpcov.

3. Prebehne kontrola umiestnenia a prítomnosti potrebných stĺpcov.

4. Do PHP objektu typu *key-value array* sa prevedie jeden riadok súboru ktorý reprezentuje jeden sektor antény, pričom kľúč je názov stĺpca a hodnota je číslo alebo textový reťazec.

5. Prebehne kontrola hodnôt pre daný riadok.

6. Objekt sa uloží do viacrozmerného poľa so štruktúrou objects[technology][cellid][sectorid].

7. Štruktúra sa prevedie do xml Štruktúry tipu VGO.

8. Xml súbor sa odošle na server kde sa uloží do databáze.

Po analýze zdrojového kódu som zistil že k prepísaniu dát dochádza v bode číslo 6 kde sa pôvodný sektor prepíše novým. Tento spôsob ukladania mal hneď 2 problémy.

1. Ak súbor obsahoval duplicitný identifikátor antén, uložená bola len druhá respektíve posledná.

2. Ak anténa obsahovala duplicitné sektory uložený bol len druhý respektíve posledný.

Tento problém som oznámil vedeniu a začal som implementáciu detekcie duplicitných sektorov. Medzičasom mi bolo oznámene, že unikátny záznam pre CDMA a LTE sa skladá z kombinácie nasledujúcich parametrov. Pre ilustráciu uvádzam len skratky atribútov pretože samotné atribúty nie sú dôležité.

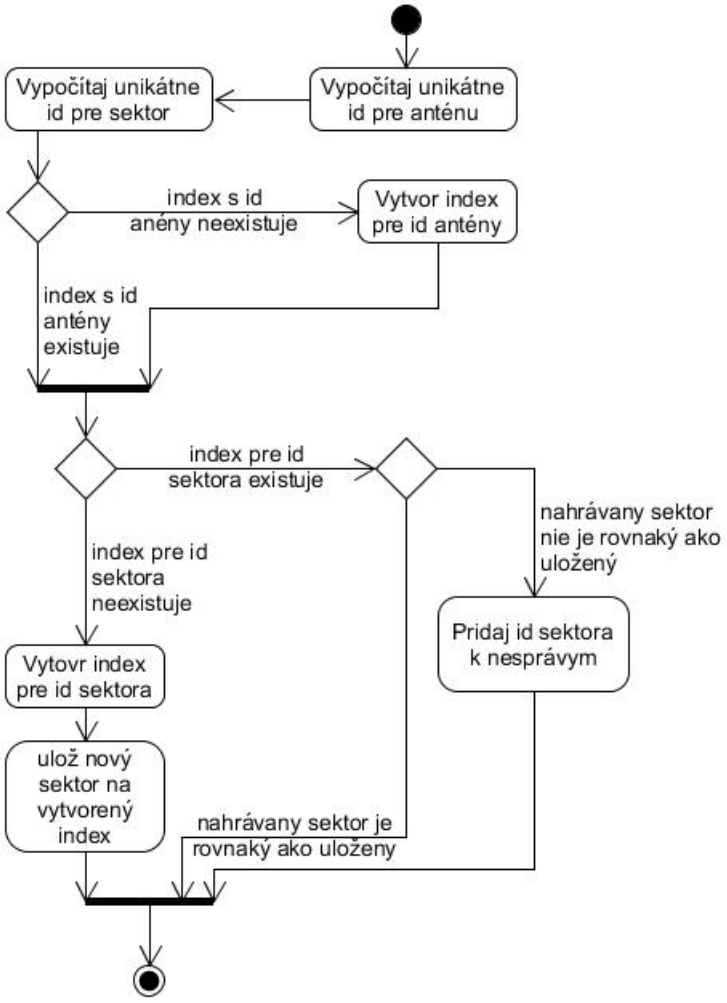
1. CDMA: cellid + MSC

2. LTE: ECI + MCCMNC + MME ip + MME port

Finálna podoba unikátneho LTE záznamu sa nezmenila. Podoba CDMA sa zmenila niekoľko krát a rozvetvila sa do niekoľkých podôb.

3.4.3 Implementácia

Implementáciu som vzhľadom k stavu projektu realizoval ako tvorbu unikátnych indexov.



Obr. 2: na obrázku môžeme vidieť ilustráciu algoritmu ktorý pri vkladaní záznamu kontroluje či neexistuje záznam s rovnakým unikátnym identifikátorom.

3.4.4 Čas vypracovania

Aj napriek jednoduchému riešeniu tejto úlohy ktorá by mi za iných okolnosti zobrala 2-3 dni som na tejto úlohe strávil značne viac času kvôli čakaniu a prepisovaniu kódu.

4 Vypracované úlohy spojené s vizualizáciou dát

Po úpravách skriptu spojených predovšetkým s podporou LTE sieti prišla na rad ich vizualizácia. V projekte už existovala mapa, ktorá sa o túto funkcionalitu starala. Mapa mala implementovaných niekoľko logických vrstiev z ktorých jedna slúžila na zobrazenie antén importovaných zákazníkom. Väčšina úloh, na ktorých som pracoval sa týkala práve tejto vrstvy. Úlohy spojené s vizualizáciou boli zaradené do tzv. *epicu* ktorý mal niekoľko cieľov:

- rozdeliť vrstvu reprezentujúcu antény na CDMA a LTE vrstvy,
- zabezpečiť aby boli dáta vizualizované aj pri nastavení menšieho zoom levelu bez straty plynulosti pri pohybe po mape,
- možnosť vyhľadávať jednotlivé anténny a miesta na mape,
- zlepšiť UX[2],

Keďže sa jednalo o väčšie rozšírenie globálny návrh a analýzu sme prediskutovali v rámci ostravského tímu a následne sme si rozdelili úlohy.

4.1 Vyhľadávanie antén a miest na mape

Zadanie úlohy bolo veľmi všeobecné a všetky detaily boli nechané na mojej kreativite. Mojou úlohou bolo teda umožniť zákazníkom vyhľadávanie antén a geografických lokalít.

4.1.1 Analýza a návrh

Pre interakciu a prácu s mapou bol v projekte použitý JavaScriptové rozšírenie OpenLayers. Stránka OpenLayers bola dostatočne dokumentovaná a obsahovala mnoho ukážkových riešení vďaka čomu som mohol po krátkom naštudovaní tohto zásuvného modulu prejsť na implementáciu.

Vyhľadávanie je bežná požiadavka a vďaka celkom veľkej popularite tohto zásuvného modulu som mohol využiť voľne dostupné rozšírenie slúžiace na vyhľadávanie lokalít na mape.

Po zakomponovaní rozšírenia do mapy som prešiel na základné nastavenia týkajúce sa spôsobu vyhľadávania a zdroja dát. Po otestovaní rozšírenia som prešiel na analýzu zdrojového kódu zásuvného modulu a následne rozšírenie o vyhľadávanie antén.

4.1.2 Implementácia

Pre vyhľadávanie antén som rozšíril prototyp o metódu, ktorá pomocou *XmlHttp* požiadavku spúšťa metódu na webovom servery. Keďže sa zákaznícke dáta nachádzajú mimo servera OMC, ktorého súčasťou je aj mapa, musel som vytvoriť požiadavku na API druhého servera. Na servery

[2] z anglického user experience – spokojnosť užívateľa pri používaní produktu

s dátami som vytvoril funkciu, ktorá sa spustí po zavolaní príslušného http GET požiadavku. Napriek tomu, že server používa c++ a pre mňa dovtedy neznámu knižnicu libevent [14] , jednalo sa len o získanie dát z databázy a ich následne odoslanie žiadateľovi. Z tohto dôvodu som sa inšpiroval konštrukciu, ktorá slúžila na podobný účel a hlbšia analýza nebola potrebná. Po úspešnom dokončení XmlHttp požiadavku nastalo overenie prítomnosti dát. Žiadne dáta indikujú že, v databáze sa nenachádza žiadna anténa s vyhľadávaným názvom. V opačnom prípade nastáva spracovanie dát. Na začiatku sa dáta prevedú do poľa a zoradia sa vzostupne podľa vzdialenosti od stredu mapy, ktorá bola viditeľná pri potvrdení vyhľadávania. Na porovnanie vzdialeností som použil tzv. harvesine vzorec ktorého implementácia je zahrnutá v OpenLayers.

```
matching_cellsites.sort(function (a,b) {
    return this$1.distanceFromViewCenter(viewCenter, a) - this$1.
        distanceFromViewCenter(viewCenter, b)
});
```

Výpis 3: V ukážke kódu môžeme vidieť prepísanie preddefinovaného spôsobu zoradenia poľa anonymnou funkciou ktorá je založená na porovnaní vzdialeností dvoch bodov na mape pomocou harvesine vzorca.

Zoradené pole sa následne prevedie ho HTML štruktúry a spustí sa udalosť rovnako ako pri vyhľadávaní lokalít. Obe udalosti sú následne odchytené a spracované v časti kódu pracujúceho s mapou.

4.1.3 Čas vypracovania

K časti projektu zameraného na prácu s mapou som sa dostal prvý krát pri tejto úlohe. Moje znalosti JavaScriptu ani rozšírenia OpenLayers neboli veľmi hlboké a preto som na tejto úlohe strávil necelé 2 týždne.

4.2 Vizualizácia dát pri malej úroveň zväčšenia

Pôvodné riešenie vizualizácie dát bolo implementované, tak že pri menšej úroveň zväčšenia sa postupne prestali vykresľovať jednotlivé vrstvy kvôli zachovaniu UX. Mojou úlohou bolo zabezpečiť, aby boli antény viditeľné aj pri malej úroveň zväčšenia a bez strát UX.

4.2.1 Analýza a návrh

Dôvodom nízkej UX bol predovšetkým veľký prenos dát spôsobený načítavaním dát z druhého servera, ktorý sa vykonával pri každej zmene view pointu[3] a množstvo dát tiež spôsobovalo ich pomalé vykresľovanie. Pomalé vykresľovanie býva často spájané predovšetkým s faktom že JavaScript je jedno vláknový a samotné vykresľovanie je náročne na zdroje.

Preto som v prvom kroku vykonal veľmi jednoduchý a priamočiary test:

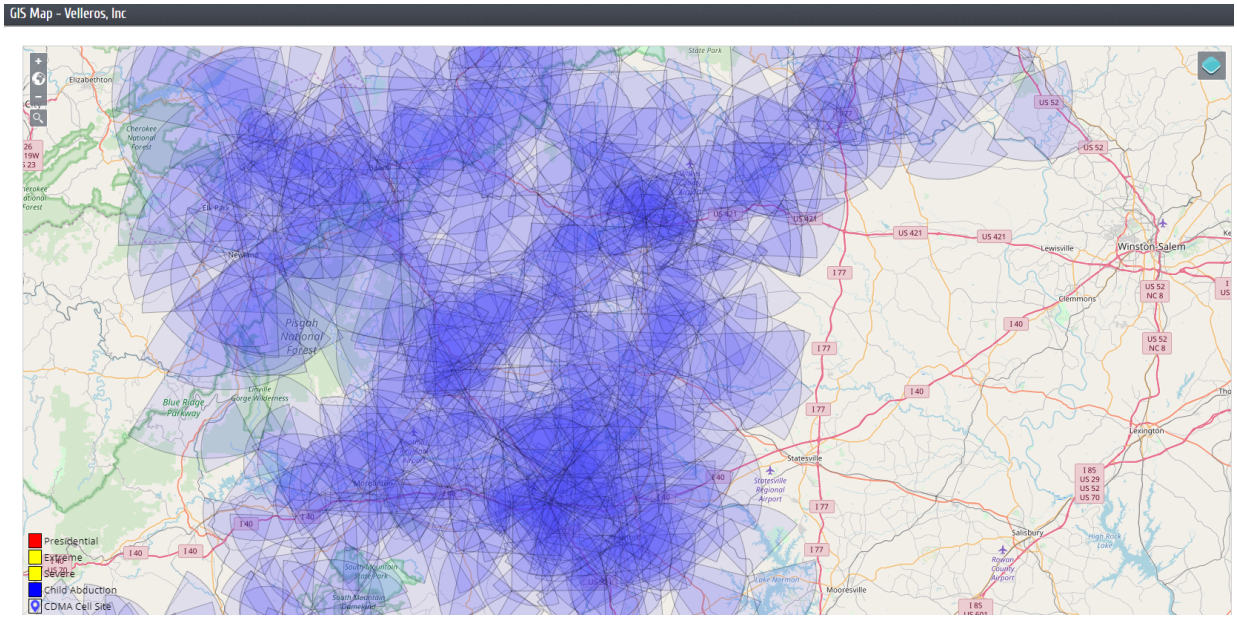
[3] view point - časť mapy viditeľná na obrazovke

1. vygeneroval a nahral som súbor s veľkým počtom antén,
2. zobrazil som oblasť mapy, kde sa nenachádzali žiadne antény a pomaly som presunul view point na miesto s anténami,
3. zobrazil som oblasť mapy, kde sa nachádzalo najviac antén s pomaly som menil view point.

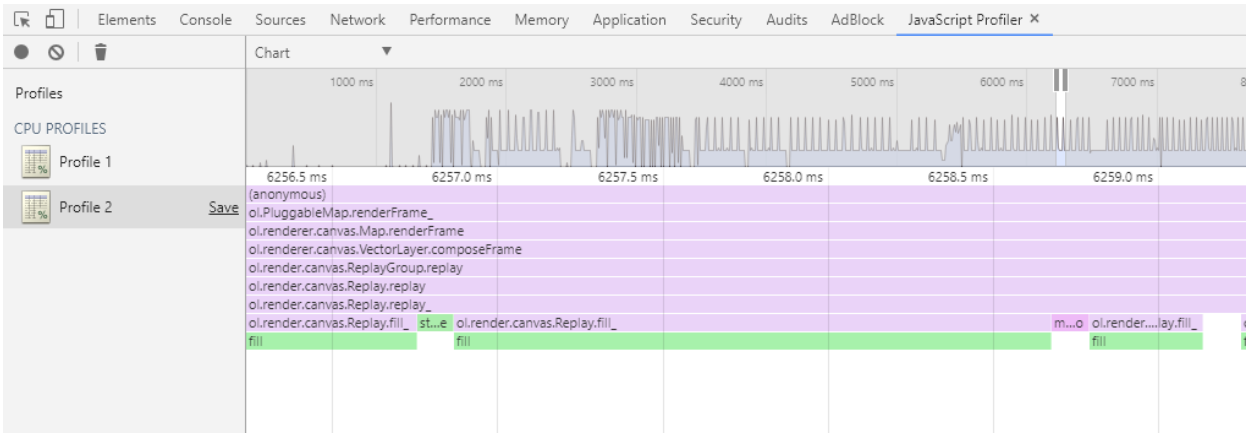
Pri tomto teste som mohol voľným okom pozorovať, že pohyb po mape je na miestach bez anténe je plynulí. S pribúdajúcimi anténami sa plynulosť znižovala až do bodu kedy UX bolo veľmi nízke. Z tohto výsledku som usúdil, že problém nastáva pri vykresľovaní jednotlivých antén a preto som vykonal podrobnejší test, na ktorý som využil JavaScript Profiler [3], ktorý je súčasťou prehliadača Google Chrome. Test prebiehal následovne:

1. spustil som nahrávanie CPU profilu,
2. zobrazil som oblasť mapy kde sa nenachádzali žiadne antény a pomaly som presunul view point na miesto s anténami,
3. záznam CPU profilu som otvoril v JavaScript profilery a urobil som analýzu z rôznych pohľadov,

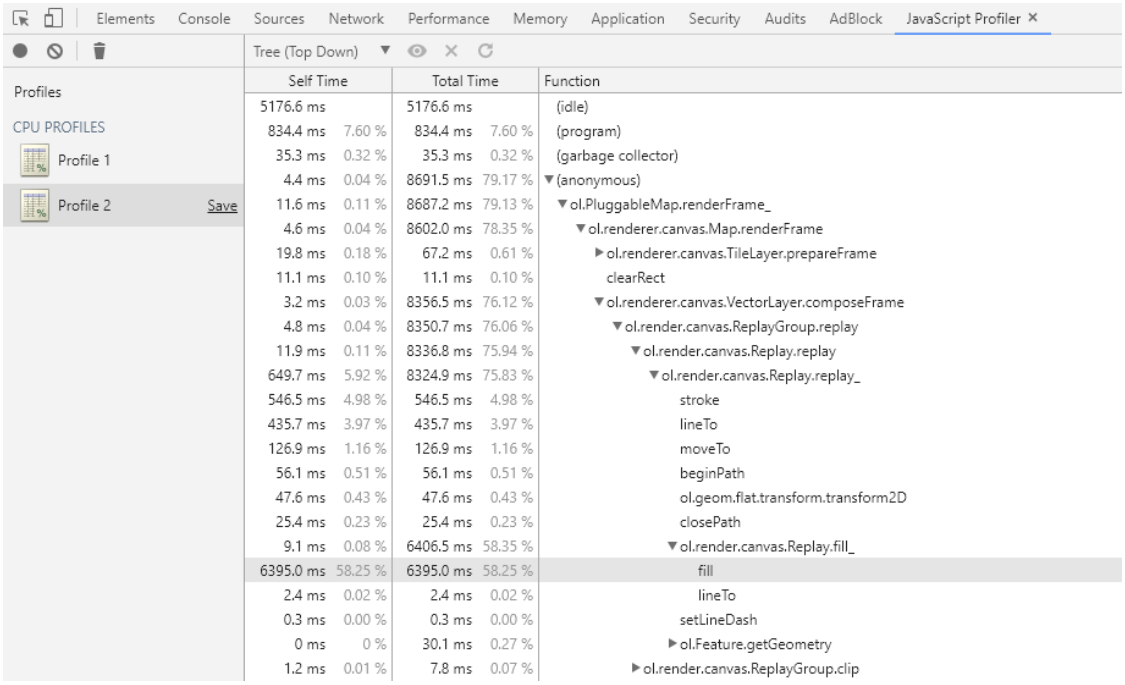
Test som previedol niekoľko krát pre minimalizovanie chýb spôsobených okolnosťami a prostredím.



Obr. 3: Na obrázku môžeme vidieť veľký počet objektov ktoré vizualizujú umiestnenie antén a smer vysielaného signálu. Túto časť mapy som použil pri testovaní.



Obr. 4: Na obrázku môžeme vidieť zachytený stav procesora počas prebiehajúceho testu. Obrázok zachytáva opakujúce sa volanie funkcie *fill* ktorá je súčasťou vykresľovania objektov a dĺžku vykonávania tejto funkcie.



Obr. 5: Na obrázku môžeme vidieť rovnaké dáta ako na obrázku 4, ale v inom formáte. Z toho formátu môžeme vyčítať koľko percent času zabrala metóda *fill*.

Z cpu profilu som vyčítal že 70-80 percent času prebiehalo samotné vykresľovanie. Z tohto času najdlhšie zabrala metóda s názvom fill, ktorá slúži na vyplnenie vykresľovaného tvaru farbou. v ďalšom kroku som využil internet a overil som si, že vykresľovanie veľkého počtu tvarov je náročne a preto sa používa tzv *"clustering"*, čo je zoskupovanie objektov do jedného objektu nazývaného taktiež clsuter, ktorý je väčšinou reprezentovaný kruhom s počtom obsiahnutých objektov. Počet objektov v jednotlivých clustroch sa v závislosti na nastavení dynamicky mení. Pri veľkej úroveň zväčšenia sa cluster vypína. So znižujúcou úroveň zväčšenia sa objekty zoskupujú

do menšieho počtu clustrov až nakoniec vznikne cluster obsahujúci všetky objekty. Pre správnu funkcionalitu potrebuje clsuter všetky dáta v čase jeho vytvorenia, vďaka čomu sa znížil prenos dát medzi servermi a spracovanie prebehlo len raz vo webworkery [1] a znížilo sa vyťaženie hlavného skriptu.

Na clustering som sa rozhodol použiť OpenLayers plugin sa názvom SuperCluster [2]. SuperClusteru som potreboval poskytnuť dáta. Keďže mapa sa využívala aj na ine účely ako zobrazovanie antén použil som Web Worker API . Externé JavaScript skripty nemajú prístup k rozšíreniu OpenLayers preto web worker sa staral iba o získanie jednotlivých sektorov antén a ich zoskupovanie do objektov reprezentujúcich jednotlive antény. Vďaka jednoduchej implementácii a veľmi dobrej dokumentácii som využil komunikačný mechanizmus na posielanie priebehu spracovania medzi webworkerom a hlavným skriptom.

Po načítaní a odovzdaní antén web workerom bolo potrebné antény načítať do SuperClusteru a pri každej zmene view pointu prepočítať, ktoré antény sa majú zobraziť osobitne a ktoré sa majú zahrnúť do clusterov. Túto funkcionalitu som pridal na koniec hlavnej vykresľovacej funkcie, ktorá sa aktivovala pri zmene view pointu.

Nakoniec som doladil základne nastavenia clsuteru a funkcionalitu som otestoval.

4.2.2 Implementácia

V prvom kroku som si vytvoril novy JavaScript súbor, ktorý slúžil ako webworker a implementoval som základnú komunikáciu s hlavným skriptom.

```
self.addEventListener('message', function (e) {
    var data = e.data;
    switch (data.cmd) {
        case 'initializeClusters':
            self.postMessage({status: 'Downloading network data ...'});
            initializeClusters();
            break;
        default:
            self.postMessage('Unknown command: ' + data.msg);
    }
}, false);
```

Výpis 4: V ukážke zdrojového kódu môžeme vidieť inicializáciu poslucháča udalosti a ich spracovanie na základe parametra *cmd*.

Podobným spôsobom som pridal event listener do hlavného skriptu

```
var worker = new Worker('/js/map/networkLayerLoader.js');
worker.addEventListener('message', function (e) {
    if (e.data.hasOwnProperty('status')) {
```

```
        $('#network-loader').text(e.data.status);
    }
    }, false);
worker.postMessage({cmd: 'initializeClusters'});
```

Výpis 5: V ukážke zdrojového kódu môžeme vidieť podobnú inicializácia ako v ukážke zdrojového kódu 4 a následne odoslanie správy webworkeru s parametrom cmd a hodnotou initializeClusters.

Ďalej som využil XMLHttp požiadavok pre získanie dát uložených v databáze na ďalšom servery. Na získanie dát som použil funkciu, ktorú som implementoval v úlohe 4.1.2 pre vyhľadávanie miest na mape. Načítané dát prebiehalo iteráciou záznamov. V cykle sa prešli a označili záznamy, ktoré mali rovnaký unikátny index 3.4.2 a následne sa dáta s rovnakým indexom uložili do objektu reprezentujúceho jednu anténu. Po načítaní každej antény sa do hlavného skriptu poslala správa s počtom spracovaných antén a po spracovaní všetkých antén pre jednotlivé technológie sa poslali dáta a následne sa zaradili k objektom, ktoré slúžili ako zdroj pre clustering.

Po načítaní všetkých dát som využil API SuperClusteru pre nastavenie základnej konfigurácie a načítanie zdrojových objektov. Ďalej som vytvoril funkciu, ktorá pri každej zmene view pointu znova prepočítala a vykreslila jednotlivé clustery alebo samotné antény.

4.2.3 Čas vypracovania

V tejto časti som sa zoznámil s nástrojmi pre profilovanie pamäte ktoré mi pomohli pri testovaní. Ďalej som si musel naštudovať problematiku Web Worker API a SuperClusteru. Tato úloha mi preto zobrala ďalší týždeň.

4.3 Zlá UX pri veľkej hustote antén

Clustering vyriešil problém so zlou UX pri malej úroveň zväčšenia. Ďalší problem s UX sa vyskytoval v prípade veľkej hustoty antén. Veľký počet antén spôsoboval, že sa mapa stala neprehľadnou a v niektorých prípadoch sa prekrývalo viac ako 10 antén a z výsledného zobrazenia 6 sa nedali vyčítať relevantné informácie. Mojou úlohou bolo pridať možnosť prepnutia sa do režimu, v ktorom budú označené len centrá antén a po ich rozkliknutí sa zobrazia všetky sektory.

4.3.1 Analýza a Návrh

Pre možnosť prepínania režimu som sa rozhodol rozšíriť funkcionalitu panela, ktorý sa využíval na vypínanie a zapínanie jednotlivých vrstiev. Keďže tento panel automatický vytváral jednotlivé možnosti z vrstiev prítomných v mape túto funkcionalitu som musel dopísať. Vytvoril som preto dve kontrolky typu *radio*:

1. Antény
2. Sektory

V režime 1. sú zobrazené len stredy antén a v režime 2. celé antény s obrysmi všetkých sektorov. Pri zmene režimu zobrazenia sa museli automaticky zmeniť vizuálne štýly antén. Taktiež som musel zariadiť, aby bolo možne v režime 1. po kliknutí na anténu zmeniť štýl pre túto anténu do režimu 2 a pri kliknutí mimo oblasť tejto antény naspäť do režimu 1. Pri tomto zobrazení som sa rozhodol zvýrazniť obrysy sektorov pre lepšiu vizualizáciu.

Zvýraznenie sektorov som sa rozhodol pridať tiež do poslednej fázy vyhľadávania. Čiže keď sa view point presunul na oblasť, kde sa nachádzala anténa, anténa sa automatický zvýraznila a v prípade, že bol zvolený režim 1 prepol sa taktiež režim pre túto anténu do režimu 2.

4.3.2 Implementácia

Keďže kolega z ostravského tímu v nedávnej dobe implementoval funkcionalitu podobnú rozšíreniu panela pre zapínanie a vypínanie vrstiev využil som jeho znalosti a poprosil som ho o implementáciu tejto funkcionality, aby som mohol efektívne využiť čas a naštudovať si problematiku dynamického menenia štýlu OpenLayers objektov.

V prvom kroku som si pripravil jednotlivé štýly ktoré reprezentovali jednotlivé stavy:

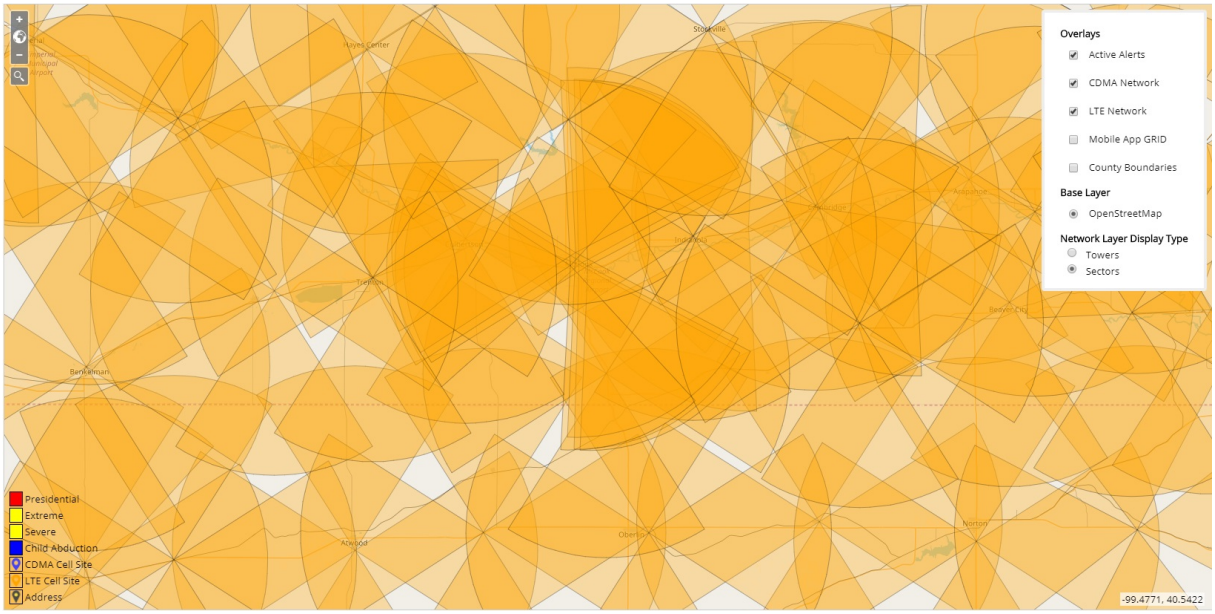
1. anténa
2. sektor
3. sektorZvolený

Následne som prepísal funkciu, ktorá sa spustila pri vykresľovaní každého objektu, v mojom prípade pri prekreslení každej antény. V tomto bode som zistil, že nemám prístup k jednotlivým objektom pretože sa nachádzajú vo vnútri superClustera. Tento problém som vyriešil vytvorením vrstvy, do ktorej sa ukladali po prekreslení clustera všetky menšie clustery ako aj jednotlivé antény, ktorým som pridal informáciu o aktuálnom režime zobrazenia. S pripravenými štýlmi a objektmi som implementoval funkcionalitu, ktorá vybrala daný štýl na základe vlastností vykresľovanej antény. Pri následnom testovaní sa zmeny štýlu neprejavovali, respektíve antény sa prestali vykresľovať. Po následnej analýze som odhalil problém, ktorý bol spôsobený použitím zlej kombinácie štýlu a geometrie objektu. Na základe tohto zistenia som musel pridať do web workera 4.2.1 kolekciu, ktorá obsahovala geometrie zodpovedajúce jednotlivým štýlom.

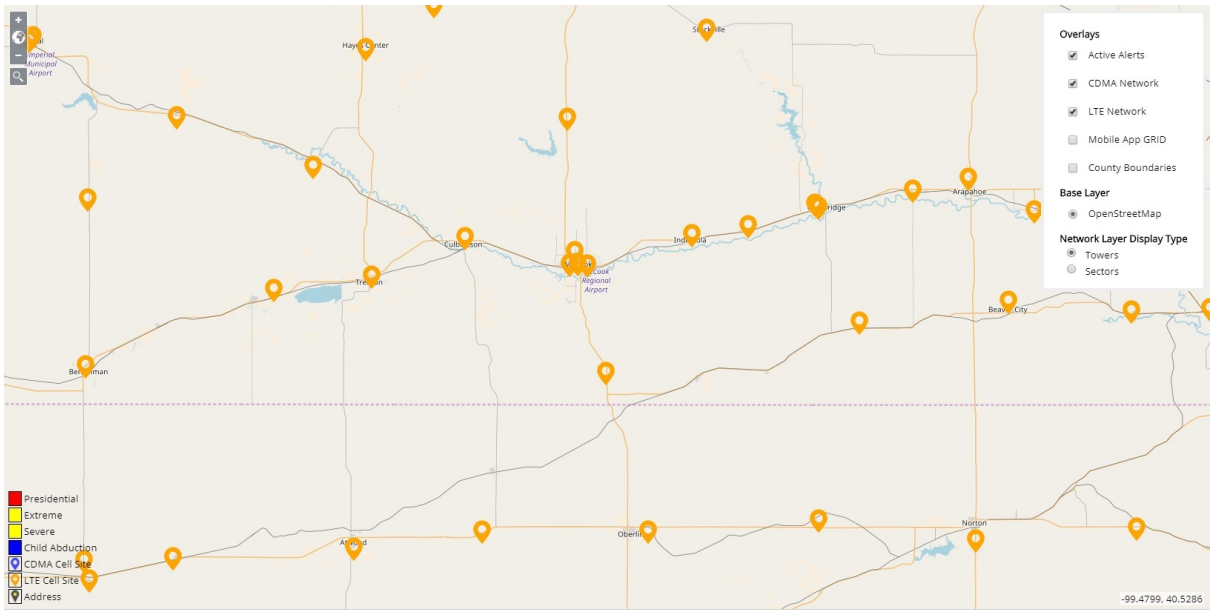
Funkčné prepínanie medzi jednotlivým režimami som prepojil s panelom pre zapínanie a vypínanie vrstiev. Keďže kolega mi pripravil funkciu, ktorá sa spustila pri zmene režimu, jediné čo som musel urobiť bolo na základe nového režimu aktualizovať jednotlivým anténam režim zobrazenia.

Následne som prešiel na zmenu režimu po kliknutí na anténu. Vďaka aktuálnej implementácii OpenLayers bola táto zmena veľmi jednoduchá. Keďže objekt reprezentujúci mapu vysielal udalosť s miestom, na ktoré bolo kliknuté zaregistroval som nový poslucháč udalosti tejto udalosti a pripojil som k nemu funkciu, ktorá odfiltrovaná všetky antény ktoré prekrývali toto miesto a poslednú vykreslila a uložila sa informácia o tom že daná anténa je zvýraznená. Pri následnom

kliknutí sa overilo, či je nejaká anténa zvýraznená a prípadne sa zvýraznenie zrušilo. Samotné vykreslenie bolo postavené na tom, že pri zmene vlastnosti objektu sa automatický zavolá funkcia ktorá vykresľuje daný objekt. Na rovnakom princípe fungovalo aj zvýraznenie v poslednom kroku vyhľadávania.



Obr. 6: Na obrázku môžeme vidieť neprehľadne zobrazenie antén v oblasti s ich hustým výskytom



Obr. 7: Na obrázku môžeme vidieť rovnaké antény ako na obrázku 6 v prehľadnom režime ktorý zobrazuje len ich geografický stred

4.3.3 Čas vypracovania

Pri implementácii tejto úlohy som narazil na niekoľko spomínaných problémov a preto mi táto úloha trvalá asi týždeň.

4.4 Zobrazenie informácii o anténe

Zadanie tejto úlohy spočívalo v rozšírenom pohľade na informácie o anténe, nad ktorou sa nachádza kurzor.

4.4.1 Analýza a Návrh

Prácu s objektami reprezentujúcimi jednotivé antény som mal v čerstvej pamäti a preto som navrhol rozšírenie funkcionality o možnosť zobrazenia všetkých antén, ktoré sa nachádzajú na danom mieste, s možnosťou zvýraznenia jednotlivých sektorov pre antény typu LTE.

Podobná funkcionalita bola implementovaná pre vrstvu, ktorá zobrazovala aktívne CMAS výstrahy. Preto som najskôr zvažoval použitie rovnakého spôsobu aj pre antény. Po dôkladnejšom preštudovaní kódu som však zistil že popup okno zobrazujúce informácie je implementované pomocou natívneho JavaScript DOM modelu. Openalayers ponúkal v aktuálnej verzii vlastnú implementáciu popup okna, ktorá bola úzko spätá s objektom reprezentujúcim mapu a poskytovala zakladanú funkcionalitu, ktorú bolo v natívnom JavaScripte potrebné do-implementovať. Keďže použitie implementácie od Openalyers navyše zvyšovalo čitateľnosť kódu a znižovalo dĺžku kódu, rozhodol som sa pre toto riešenie.

4.4.2 Implementácia

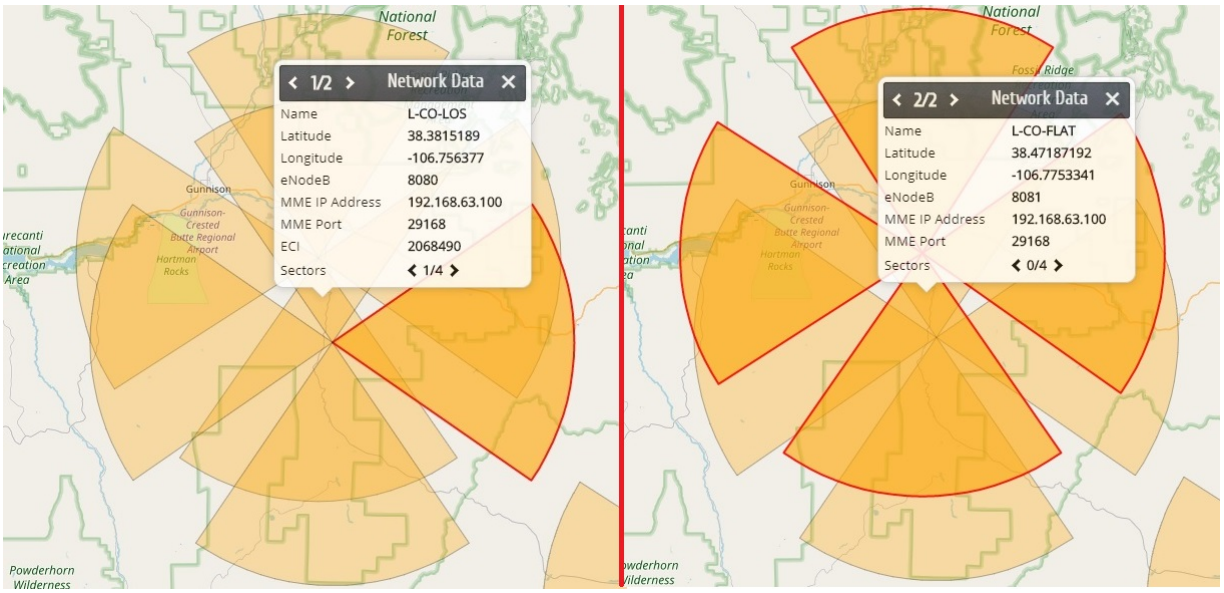
V prvom kroku som si pripravil objekt reprezentujúci HTML element pre správu. Tento HTML element som vložil do objektu reprezentujúceho okno popup správy a samotný objekt som pripojil k vrstve mapy, ktorá reprezentovala všetky podobné objekty.

Popup oknu som nastavil preddefinovanú pozíciu na *'undefined'*, čo je jedno z kľúčových slov jazyka JavaScript a zabezpečoval, že popup sa nezobrazoval. Pre zobrazenie popup okna som vytvoril funkciu ktorú som pripojil na udalosť, ktorá sa vyvolávala pri každom pohybe kurzora. V prípade, že sa pod kurzorom nachádzali nejaké antény alebo CMAS výstrahy dynamicky sa vytvoril HTML element, v ktorom boli informácie o danej anténe alebo CMAS výstrahe.

Objekty s vlastnosťami antén neobsahovali tieto podrobnosti preto som musel znova rozšíriť webworker 4.2.1 o zahrnutie týchto informácii. Pre LTE antény som taktiež pridal novú kolekciu, ktorá reprezentovala geometrie jednotlivých sektorov antény.

Aby bolo možne prechádzať všetky platne objekty nad ktorým sa nachádzal kurzor, rozšíril som funkciu, ktorá sa spustila pri kliknutí na mapu. Funkcia najskôr overila či je popu okno zamknuté - tento zámok spôsoboval, že sa uzamklo okno so všetkými platnými objektmi a pri pohybe kurzoru sa prestalo aktualizovať, vďaka čomu mohol užívateľ prepínať medzi jednotlivými

objektmi a pri LTE antách, taktiež medzi sektormi. Na prepínanie sektorov som si vytvoril 2 funkcie, ktoré zobrazili buď nasledujúcu alebo predchádzajúcu anténu. Tieto funkcie prechádzali objekty, ktoré boli uložené po uzamknutí popup okna a dynamicky menili HTML na základe ich vlastnosti. Rovnakým spôsobom som implementoval prepínanie medzi sektormi



Obr. 8: Na obrázku môžeme vidieť dva pohľady na anténu. V pravej časti obrázku sa nachádza anténa zobrazujúca anténu ako celok. V ľavej časti obrázka sa nachádza anténa so zvýrazneným sektorom číslo 1 a popup oknom rozšíreným o atribút ECI ktorý je unikátny pre každý sektor.

4.4.3 Čas vypracovania

V poslednej úlohe som narazil na podobné problémy ako v predchádzajúcich úlohách a pracoval som so svojim zdrojovým kódom vďaka čomu mi vypracovanie trvalo 4 dni.

5 Teoretické a praktické vedomosti využite v priebehu praxe

V praxi som využil predovšetkým všeobecný prehľad technológii a postupov spojených s vývojom softvéru. Z môjho pohľad každý predmet absolvovaný na VŠB prinesie kúsok znalostí potrebných k práci programátora.

Menovite by som chcel vyzdvihnúť predmety ako Úvod do databázových systémov, vďaka ktorému sme nemal len dostatočné znalosti pri vytváraní príkazov pre získavanie dát z databáze, ktoré sú nutnosťou väčšiny programov, ale dovolil by som si povedať, že hĺbka tohto predmetu mi poskytla do istej miery nadhľad nad danou problematikou.

Ďalší predmet, ktorý by som rád vyzdvihol je Úvod do teoretickej informatiky a predovšetkým znalosti spojené s tvorením regulárnych výrazov.

Ako nevyhnutné predmety považujem Algoritmy 1 a Algoritmy 2, v ktorých som získal znalostí potrebné k tvoreniu zložitejších algoritmov a vyvarovania sa príliš veľkej časovej a pamäťovej náročnosti.

Ako základný kameň návrhu a tvorby rozsiahlejších programov považujem predmety Programovanie 1 a Programovanie 2, predovšetkým projekty spojené s týmito predmetmi. Vďaka rozsiahlosti a zameraniu týchto projektov som mal dostatočný základ v oblasti členenia programu do menších funkcii a celkov obvykle reprezentovaných triedami.

6 Teoretické a praktické vedomosti chýbajúce pri vykonávaní praxe

Do prvej skupiny by som zaradil znalosti ktoré mi síce chýbali, ale ich nadobudnutie je spojené predovšetkým so skúsenosťami z praxe, ktoré sa obtiažne dosahujú na akademickej pôde. Medzi tieto vlastnosti patrí predovšetkým spolupráca v tíme, časoví odhad splnenia úloh a hlbší návrh.

Ďalšia vedomosť, ktorá mi chýbala v priebehu praxe bola práca s operačným systémom Linux. Keďže Fakulta Elektroniky a Informatiky poskytuje predmety zamerané na túto problematiku v priebehu druhej časti praxe som si zvoli predmet Správa operačných systémov, ktorý mi vo veľkej miere objasnil a rozšíril vedomosti o systéme Linux, a preto by som ho zaradil k predmetom ktoré mi chýbali ale dodatočne mi pomohli. Verím, že mi znalosti nadobudnuté v tomto predmete budú užitočne v ďalšej programátorskej praxi.

Znalosť, ktorá mi chýbala predovšetkým na začiatku praxe bola znalosť systému pre správu verzii kódu. Z môjho pohľadu táto znalosť patrí k elementárnym nie len v praxi, ale aj pri tvorbe väčších projektov vrámci výuky. Použitie systému pre správu verzii kódu bola jedna z podmienok projektu vypracovaného na predmete Tvorba Mobilných Aplikácii 2 kde som využil znalosti nadobudnuté počas praxe a nie naopak.

Ako menšie mínus beriem prevahu predmetov zameraných na jazyky so statickou typovou kontrolou. V rámci praxe som sa stretol s jazykmi ako PHP či JavaScript, ktoré majú dynamickú typovú kontrolu. Fungovanie a princíp týchto jazykov som sa musel dodatočne doučiť. V rámci praxe som sa nepriamo stretol s programovaním riadeným udalosťami, tento štýl ma veľmi zaujal a uvítal by som predmet/časť predmetu zameranú na túto techniku.

7 Záver

Vykonanie bakalárskej práce formou odbornej praxe vo firme je výborná príležitosť. Keďže ma vždy zaujímalo, či znalosti získane na akademickej pôde odpovedajú každodennej praxi, túto možnosť som využil.

Súčasťou praxe nebolo len programovanie samotné. Počas tejto doby som si vyskúšal niekoľko programovacích jazykov, z ktorých mi každý rozšíril a zmenil pohľad na písanie a návrh softvéru.

Vyskúšal som si prácu v tíme, čo je z môjho pohľadu jeden najdôležitejších aspektov moderného programovania. Prakticky som uplatnil akademicky získane znalosti o agilnom prístupe pri firmou uplatňovanej metodike s názvom SCRUM.

Ako veľké plus beriem možnosť pracovať pre americkú firmu. Vďaka tejto možnosti som si nielen zlepšil angličtinu, ale zároveň som si vyskúšal komunikáciu v cudzom a pre mňa doteraz neznámom prostredí.

Literatura

[1] MDN Web Docs. (2018). Web Workers API. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API [cit. 2018-03-22]

[2] GitHub. (2018). mapbox/supercluster. [online] Available at: https://github.com/mapbox/supercluster [cit. 2018-03-20]

[3] Google Developers. (2018). Tools for Web Developers.[online] Available at: https://developers.google.com/web/tools/chrome-devtools/rendering-tools/ [cit. 2018-03-22]

[4] En.wikipedia.org. (2018). Scrum (software development). [online] Available at: https://en.wikipedia.org/wiki/Scrum_(software_development) [cit. 2018-03-22]

[5] En.wikipedia.org. (2018). Google Hangouts. [online] Available at: https://en.wikipedia.org/wiki/Google_Hangouts [cit. 2018-03-27]

[6] Pivotaltracker.com. (2018). Features. [online] Available at: https://www.pivotaltracker.com/features [cit. 2018-03-27]

[7] En.wikipedia.org. (2018). GitHub. [online] Available at: https://en.wikipedia.org/wiki/GitHub [cit. 2018-03-27]

[8] En.wikipedia.org. (2018). Git. [online] Available at: https://en.wikipedia.org/wiki/Git [cit. 2018-04-20]

[9] En.wikipedia.org. (2018). Federal Emergency Management Agency. [online] Available at: https://en.wikipedia.org/wiki/Federal_Emergency_Management_Agency [cit. 2018-04-20]

[10] SourceTree. (2018). Sourcetree | Free Git GUI for Mac and Windows. [online] Availableat:https://www.sourcetreeapp.com/ [cit. 2018-04-20]

[11] Zend, a. (2018). Home - Zend Framework. [online] Framework.zend.com. Availableat:https://framework.zend.com/ [cit. 2018-04-20]

[12] Php.net. (2018). PHP: preg-match - Manual. [online] Available at: http://php.net/manual/en/function.preg-match.php [cit. 2018-04-20]

[13] Phpliveregex.com. (2018). PHP Live Regex. [online] Available at: http://www.phpliveregex.com/ [cit. 2018-04-20]

[14] Libevent.org. [cit. 2018-03-20]. libevent. [online] Available at: http://libevent.org/ [cit. 2018-04-20]